1. A brief description of the notable obstacles you overcame.

Fist obstacle that I faced was to generate a valid MegaMillionTicket. Even though it was

stated that we do not have to worry about this issue as the tester will only test with valid tickets,

I was still quite willing to have a sort of response or check for ticket validation. Therefore, I

added three private functions to check the tickets: withinRange of specs, had no duplicate for the

first five balls; once a bad ticket was generated, the main will use an exception statement to catch

the bad ticket.

Then, I spent some time making a valid lottery from the randomNumer generator as again check

for range and duplicates. Since it would better for us to keep the functions private, I simply

copied and pasted the previous codes for the private functions from MegaMillionTicket.cpp for

those purposes. The idea here was to check for range for all balls and check for a duplicate for

ball 1-5, if there is a duplicate, then run the randomNumber generator again till there is not for

the next ball.

1. A list of the test data that could be used to thoroughly test your functions, along with the reason for each test.
2. Ticket(1,2,3,4,5,6); lottery (1,2,3,4,5,6) for all matching with valid ticket/ lottery
3. Ticket(1,2,3,4,5,6); lottery (1,2,3,4,5,12) for 5 matching with valid ticket/ lottery
4. Ticket (1,1,1,1,1,1) for invalid ticket
5. Ticket(1,2,3,4,5,1) for valid ticket with duplicate megaball
6. Lottery (1,2,3,4,5,1) for valid lottery with duplicate megaball
7. MegaMillionsTicket mTickets() for default constructor
8. Ticket (1,2,3,4,5,6); lottery (5,4,3,2,1,6) for valid and all matching with not matching positions
9. Ticket (1, 2, 3, 4, 5, 6); lottery (8, 10, 11, 12, 2, 6) with one matching plus megaball
10. Tested with code example for testing on assignment for getball and random functions